

Introducción a R

José Ramón Berrendero

Departamento de Matemáticas

Universidad Autónoma de Madrid

Lo que vamos a ver

- **Aspectos básicos de R**
 - Tipos de objetos: vectores, matrices, listas, data frames
- **¿Cómo manejar los datos?**
 - Importar datos a R
 - Introducción a dplyr
- **¿Cómo visualizar los datos?**
 - Introducción a ggplot2
- **Herramientas de comunicación**
 - R markdown
- **Referencias**

Aspectos básicos

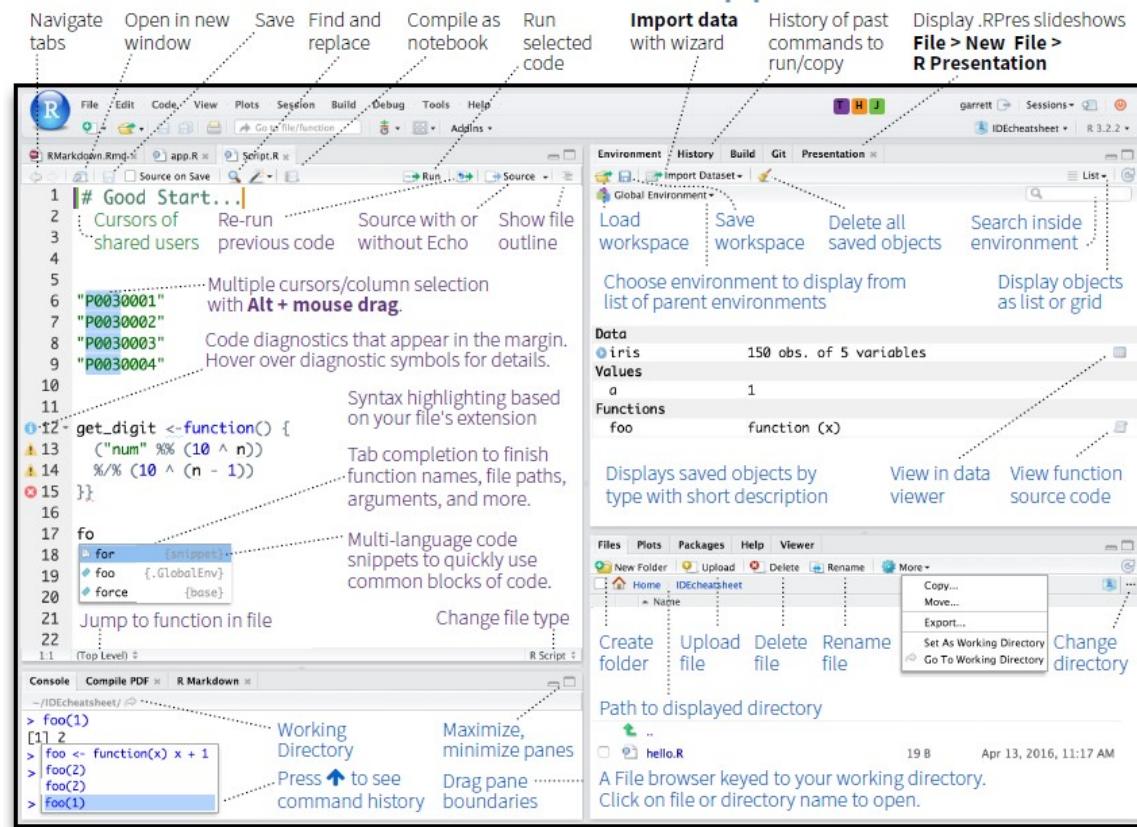
Aspectos básicos

- Sobre la mejor forma de trabajar
- Características básicas
- Algunos tipos de objetos importantes:
 - [Vectores](#)
 - [Factores](#)
 - [Matrices](#)
 - [Listas](#)
 - [Ficheros de datos \(*data frames*\)](#)

Una descripción de la interfaz

- La consola
- El editor de texto
- La ventana del entorno de trabajo
- La ventana de la ayuda, los gráficos y el explorador

Una descripción de la interfaz



Sobre la mejor forma de trabajar

- Ficheros .R (texto con código) y ficheros .RData (objetos de R)
- Mantener separados los ficheros de proyectos diferentes
- Para ello, abrir un proyecto nuevo cada vez que se empieza un trabajo
- Escribir código en lugar de trabajar de forma interactiva
- Mejor guardar el código que los resultados (reproducibilidad)

Características básicas

- Orientado a objetos
- Distingue mayúsculas y minúsculas
- Los nombres de objetos empiezan por una letra
- Operador de asignación: `x <- 10`
- Combinar objetos: `x <- c(2, 3, 4)`
- Lista de objetos en el área de trabajo `ls()`
- Borrar objetos: `rm()`
- Obtener ayuda: `help()` o `help.search()`

Vectores: comandos para generarlos

```
1:5
```

```
## [1] 1 2 3 4 5  
  
seq(2, 10, 2)  
  
## [1] 2 4 6 8 10  
  
rep(c(1,3), 4)  
  
## [1] 1 3 1 3 1 3 1 3
```

Vectores: reciclaje

R automáticamente *recicla* (repite los valores) hasta llegar a la dimensión requerida:

```
3 + c(1,2,3,4)
```

```
## [1] 4 5 6 7
```

```
c(1,2) + c(7,8,9,10)
```

```
## [1] 8 10 10 12
```

Vectores: filtrado

Cómo seleccionar algunas de las coordenadas:

```
x <- (1:5)^2  
x[2]
```

```
## [1] 4
```

```
x[c(2,5)]
```

```
## [1] 4 25
```

```
x[-c(2,5)]
```

```
## [1] 1 9 16
```

Vectores: filtrado

Cómo seleccionar los elementos que cumplen cierta condición:

```
x <- (1:5)^2  
x[x > 15]
```

```
## [1] 16 25
```

```
which(x>15)
```

```
## [1] 4 5
```

Factores

Son las estructuras que se utilizan para manejar las variables cualitativas en los análisis estadísticos.

Se pueden contemplar como un vector al que se añade un poco más de información consistente en los distintos valores presentes en el vector, llamados *niveles*.

Se crean usando el comando `factor()`:

```
respuesta <- c('si', 'no', 'si', 'no', 'no', 'no', 'si')
respuesta <- factor(respuesta)
respuesta

## [1] si no si no no no si
## Levels: no si
```

Matrices

Las matrices **son también vectores** pero con dos atributos adicionales: número de filas y número de columnas

Los vectores **no son matrices con una fila o con una columna**

Para crear una matriz:

```
matrix(1:3, nrow = 2, ncol = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    2
## [2,]    2    1    3
```

Las matrices son también vectores

```
x <- matrix(2:10, 3, 3)
```

```
x
```

```
##      [,1] [,2] [,3]
## [1,]     2     5     8
## [2,]     3     6     9
## [3,]     4     7    10
```

```
x[x>6]
```

```
## [1] 7 8 9 10
```

Seleccionar submatrices

```
x[, 1, 2]
```

```
## [1] 5
```

```
x[, , 3]
```

```
## [1] 8 9 10
```

```
x[3, -1]
```

```
## [1] 7 10
```

```
x[c(1, 2), ]
```

```
##      [,1] [,2] [,3]
## [1,]     2     5     8
## [2,]     3     6     9
```

Algunas operaciones útiles con matrices

```
A = matrix(1:4,2,2) ; B = matrix(5:8,2,2)
```

- Producto matricial:

```
A %*% B
```

```
[,1] [,2]  
[1,] 23 31  
[2,] 34 46
```

- Determinante:

```
det(A)
```

```
[1] -2
```

- Producto componente a componente:

```
A * B
```

```
[,1] [,2]  
[1,]    5   21  
[2,]   12   32
```

- Traspuesta:

```
t(A)
```

```
[,1] [,2]  
[1,]    1   2  
[2,]    3   4
```

- Extraer la diagonal:

```
diag(A)
```

```
[1] 1 4
```

- Inversa:

```
solve(A)
```

```
 [,1] [,2]
[1,] -2 1.5
[2,] 1 -0.5
```

- Resolver un sistema de ecuaciones lineales $Ax = b$:

```
b = c(1,3)
solve(A,b)
```

```
[1] 2.5 -0.5
```

- Autovalores y autovectores:

```
eigen(A) # Devuelve una lista
```

```
eigen() decomposition
$values
[1] 5.3722813 -0.3722813

$vectors
[,1]      [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

Listas

Una lista es un vector de objetos de tipos distintos, que conviene agrupar en la misma estructura

Es importante comprenderlas porque muchas funciones de R devuelven los resultados en forma de lista

Para crearlas se usa el comando `list`:

```
milista <- list(nombre='Pepe',no.hijos=3,  
edades.hijos=c(4,7,9))  
str(milista)
```

```
## List of 3  
## $ nombre      : chr "Pepe"  
## $ no.hijos    : num 3  
## $ edades.hijos: num [1:3] 4 7 9
```

Extraer información de una lista

Cómo extraer la información de una lista:

```
milista$nombre
```

```
## [1] "Pepe"
```

```
milista[[1]] # primera alternativa
```

```
## [1] "Pepe"
```

```
milista[['nombre']] # segunda alternativa
```

```
## [1] "Pepe"
```

```
milista$edades.hijos[2]
```

```
## [1] 7
```

Algunas cuestiones

- ¿Cuál es la diferencia entre `milista[1]` y `milista[[1]]`?
- ¿Cuál es el resultado de `milista[2:3]`?
- ¿Y el resultado de `milista[[2:3]]`?

Data frames

La estructura más importante en R. Son los ficheros de datos

Intuitivamente son matrices con entradas de distintos tipos. Técnicamente es una lista formada por vectores (heterogéneos) de la misma longitud

Para crearlos se utiliza el comando `data.frame`:

```
x <- 7:9
y <- c('a', 'b', 'c')
mifichero <- data.frame(edad=x, grupo=y)
mifichero

##   edad grupo
## 1    7     a
## 2    8     b
## 3    9     c
```

Data frames

Admiten comandos para matrices y para listas:

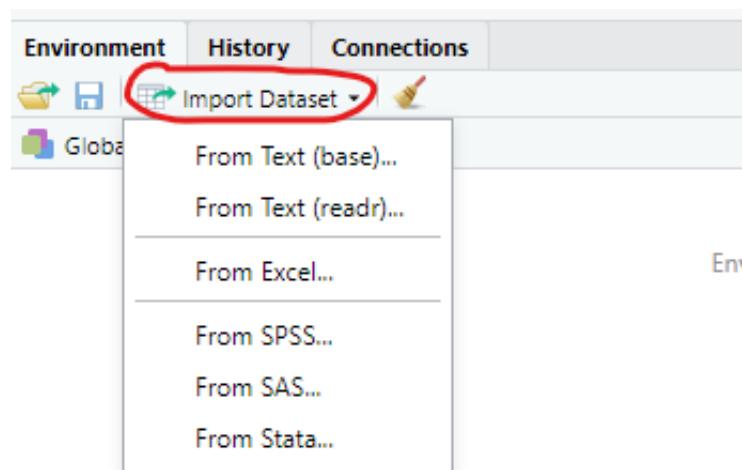
```
mifichero$edad
```

```
## [1] 7 8 9
```

```
mifichero[1,]
```

```
##   edad grupo
## 1     7     a
```

Importar datos a R (desde RStudio)



Ejercicios:

- Lee con `R` el fichero excel (renta y fracaso escolar en la Comunidad de Madrid) que puedes encontrar [en este enlace](#)
- Lee con `R` el fichero de texto (notas de colegios de la Comunidad de Madrid) que puedes encontrar [en este enlace](#)

Manejo de datos con dplyr

Una introducción a dplyr

Un paquete del [tidyverse](#)

Incluye las acciones más comunes que se realizan sobre un conjunto de datos:

- [Contar: count](#)
- [Seleccionar filas: filter](#)
- [Seleccionar columnas: select](#)
- [Ordenar: arrange](#)
- [Sintaxis en cadena \(pipes\)](#)
- [Añadir nuevas variables: mutate](#)
- [Resumir: summarise](#)

Características generales

- El primer argumento siempre es un *data.frame*
- El resto de argumentos indican lo que queremos hacer con el *data.frame*
- El resultado siempre tiene también la estructura de *data.frame*

Una introducción a dplyr

Seleccionamos 15 observaciones (5 de cada especie) del fichero `iris` a la manera *tradicional*:

```
lirios <- iris[c(1:5,51:55,101:105),]  
head(lirios)
```

	## Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa
## 4	4.6	3.1	1.5	0.2	setosa
## 5	5.0	3.6	1.4	0.2	setosa
## 51	7.0	3.2	4.7	1.4	versicolor

Usaremos `lirios` como fichero de datos en los ejemplos que siguen

Antes de empezar cargamos `dplyr`:

```
library(dplyr)
```

Contar: count

¿Cuántos lirios hay de cada especie?

```
count(lirios, Species)
```

```
##      Species n
## 1      setosa 5
## 2 versicolor 5
## 3 virginica 5
```

Seleccionar filas: filter

Selecciona todos los lirios de la especie *setosa*:

```
filter(lirios, Species == 'setosa')
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         5.1       3.5        1.4       0.2  setosa
## 2         4.9       3.0        1.4       0.2  setosa
## 3         4.7       3.2        1.3       0.2  setosa
## 4         4.6       3.1        1.5       0.2  setosa
## 5         5.0       3.6        1.4       0.2  setosa
```

Seleccionar filas: filter

Selecciona todos los lirios de la especie *setosa* o *virginica*:

```
filter(lirios, Species=='setosa' | Species == 'virginica')
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1        3.5       1.4        0.2   setosa
## 2          4.9        3.0       1.4        0.2   setosa
## 3          4.7        3.2       1.3        0.2   setosa
## 4          4.6        3.1       1.5        0.2   setosa
## 5          5.0        3.6       1.4        0.2   setosa
## 6          6.3        3.3       6.0        2.5 virginica
## 7          5.8        2.7       5.1        1.9 virginica
## 8          7.1        3.0       5.9        2.1 virginica
## 9          6.3        2.9       5.6        1.8 virginica
## 10         6.5        3.0       5.8        2.2 virginica
```

Seleccionar filas: filter

Selecciona los lirios de la especie *setosa* cuya longitud de sépalo es inferior a 5 mm:

```
filter(lirios, Species == 'setosa' & Sepal.Length < 5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         4.9      3.0       1.4      0.2    setosa
## 2         4.7      3.2       1.3      0.2    setosa
## 3         4.6      3.1       1.5      0.2    setosa
```

Seleccionar columnas: select

Selecciona las variables relacionadas con el sépalo:

```
select(lirios, Sepal.Length, Sepal.Width)
```

```
##      Sepal.Length Sepal.Width
## 1          5.1       3.5
## 2          4.9       3.0
## 3          4.7       3.2
## 4          4.6       3.1
## 5          5.0       3.6
## 51         7.0       3.2
## 52         6.4       3.2
## 53         6.9       3.1
## 54         5.5       2.3
## 55         6.5       2.8
## 101        6.3       3.3
## 102        5.8       2.7
## 103        7.1       3.0
## 104        6.3       2.9
## 105        6.5       3.0
```

Seleccionar columnas: select

Selecciona un rango de variables:

```
select(lirios, Petal.Length:Sepal.Length)
```

```
##      Petal.Length Sepal.Width Sepal.Length
## 1          1.4        3.5       5.1
## 2          1.4        3.0       4.9
## 3          1.3        3.2       4.7
## 4          1.5        3.1       4.6
## 5          1.4        3.6       5.0
## 51         4.7        3.2       7.0
## 52         4.5        3.2       6.4
## 53         4.9        3.1       6.9
## 54         4.0        2.3       5.5
## 55         4.6        2.8       6.5
## 101        6.0        3.3       6.3
## 102        5.1        2.7       5.8
## 103        5.9        3.0       7.1
## 104        5.6        2.9       6.3
## 105        5.8        3.0       6.5
```

Seleccionar columnas: select

Selecciona todas las variables menos *Species*:

```
select(lirios, -Species)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1       3.5        1.4       0.2
## 2          4.9       3.0        1.4       0.2
## 3          4.7       3.2        1.3       0.2
## 4          4.6       3.1        1.5       0.2
## 5          5.0       3.6        1.4       0.2
## 51         7.0       3.2        4.7       1.4
## 52         6.4       3.2        4.5       1.5
## 53         6.9       3.1        4.9       1.5
## 54         5.5       2.3        4.0       1.3
## 55         6.5       2.8        4.6       1.5
## 101        6.3       3.3        6.0       2.5
## 102        5.8       2.7        5.1       1.9
## 103        7.1       3.0        5.9       2.1
## 104        6.3       2.9        5.6       1.8
## 105        6.5       3.0        5.8       2.2
```

Seleccionar columnas: select

Selecciona las variables cuyo nombre contiene *Petal*:

```
select(lirios, contains('Petal'))
```

```
##      Petal.Length Petal.Width
## 1          1.4        0.2
## 2          1.4        0.2
## 3          1.3        0.2
## 4          1.5        0.2
## 5          1.4        0.2
## 51         4.7        1.4
## 52         4.5        1.5
## 53         4.9        1.5
## 54         4.0        1.3
## 55         4.6        1.5
## 101        6.0        2.5
## 102        5.1        1.9
## 103        5.9        2.1
## 104        5.6        1.8
## 105        5.8        2.2
```

Más información sobre selección de variables

Ordenar: arrange

Ordena las filas de **menor a mayor** valor de la variable *Sepal.Length*:

```
arrange(lirios, Sepal.Length)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1         4.6      3.1        1.5       0.2  setosa
## 2         4.7      3.2        1.3       0.2  setosa
## 3         4.9      3.0        1.4       0.2  setosa
## 4         5.0      3.6        1.4       0.2  setosa
## 5         5.1      3.5        1.4       0.2  setosa
## 6         5.5      2.3        4.0      1.3 versicolor
## 7         5.8      2.7        5.1      1.9  virginica
## 8         6.3      3.3        6.0      2.5  virginica
## 9         6.3      2.9        5.6      1.8  virginica
## 10        6.4      3.2        4.5      1.5 versicolor
## 11        6.5      2.8        4.6      1.5 versicolor
## 12        6.5      3.0        5.8      2.2  virginica
## 13        6.9      3.1        4.9      1.5 versicolor
## 14        7.0      3.2        4.7      1.4 versicolor
## 15        7.1      3.0        5.9      2.1  virginica
```

Ordenar: arrange

Ordena las filas de **mayor a menor** valor de la variable *Sepal.Length*:

```
arrange(lirios, -Sepal.Length)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width     Species
## 1          7.1        3.0       5.9        2.1  virginica
## 2          7.0        3.2       4.7        1.4 versicolor
## 3          6.9        3.1       4.9        1.5 versicolor
## 4          6.5        2.8       4.6        1.5 versicolor
## 5          6.5        3.0       5.8        2.2  virginica
## 6          6.4        3.2       4.5        1.5 versicolor
## 7          6.3        3.3       6.0        2.5  virginica
## 8          6.3        2.9       5.6        1.8  virginica
## 9          5.8        2.7       5.1        1.9  virginica
## 10         5.5        2.3       4.0        1.3 versicolor
## 11         5.1        3.5       1.4        0.2    setosa
## 12         5.0        3.6       1.4        0.2    setosa
## 13         4.9        3.0       1.4        0.2    setosa
## 14         4.7        3.2       1.3        0.2    setosa
## 15         4.6        3.1       1.5        0.2    setosa
```

Ordenar: arrange

Ordena los lirios según la especie (por orden alfabético) y dentro de cada especie ordena de menor a mayor longitud del sépalo:

```
arrange(lirios, Species, Sepal.Length)
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	4.6	3.1	1.5	0.2	setosa
## 2	4.7	3.2	1.3	0.2	setosa
## 3	4.9	3.0	1.4	0.2	setosa
## 4	5.0	3.6	1.4	0.2	setosa
## 5	5.1	3.5	1.4	0.2	setosa
## 6	5.5	2.3	4.0	1.3	versicolor
## 7	6.4	3.2	4.5	1.5	versicolor
## 8	6.5	2.8	4.6	1.5	versicolor
## 9	6.9	3.1	4.9	1.5	versicolor
## 10	7.0	3.2	4.7	1.4	versicolor
## 11	5.8	2.7	5.1	1.9	virginica
## 12	6.3	3.3	6.0	2.5	virginica
## 13	6.3	2.9	5.6	1.8	virginica
## 14	6.5	3.0	5.8	2.2	virginica
## 15	7.1	3.0	5.9	2.1	virginica

Sintaxis en cadena (*pipes*)

Podemos *leer* el operador `%>%` como *entonces*

El elemento que le precede es el primer argumento del comando que le sigue

```
lirios %>%
  select(contains('Petal'))    %>%
  filter(Petal.Length > 4)    %>%
  arrange(Petal.Length)
```

```
##   Petal.Length Petal.Width
## 1          4.5         1.5
## 2          4.6         1.5
## 3          4.7         1.4
## 4          4.9         1.5
## 5          5.1         1.9
## 6          5.6         1.8
## 7          5.8         2.2
## 8          5.9         2.1
## 9          6.0         2.5
```

Añadir nuevas variables: mutate

Crea una nueva variable que corresponde al cociente entre la anchura y la longitud del pétalo (que podría cuantificar la forma del pétalo):

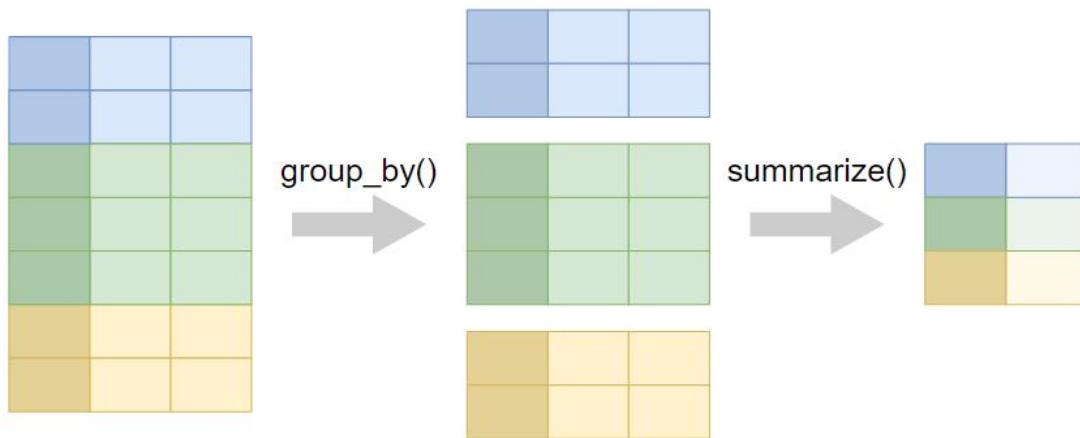
```
lirios_nuevo <-
  lirios %>%
  select(contains('Petal')) %>%
  mutate(forma = round(Petal.Width/Petal.Length, 2))

head(lirios_nuevo)
```

```
##      Petal.Length Petal.Width forma
## 1          1.4       0.2   0.14
## 2          1.4       0.2   0.14
## 3          1.3       0.2   0.15
## 4          1.5       0.2   0.13
## 5          1.4       0.2   0.14
## 51         4.7       1.4   0.30
```

Resumir: summarise

Se suele usar en combinación con `group_by`



Resumir: summarise

Calcula la media de la longitud del pétalo para los lirios de cada una de las especies

```
lirios %>%
  group_by(Species) %>%
  summarise(media_PL = mean(Petal.Length),
            varianza_PL = var(Petal.Length))
```

```
## # A tibble: 3 x 3
##   Species     media_PL varianza_PL
##   <fct>        <dbl>      <dbl>
## 1 setosa       1.4       0.0050
## 2 versicolor   4.54      0.113
## 3 virginica    5.68      0.127
```

Operaciones por columnas: across()

Calcula las medias y las desviaciones típicas de cada medida del pétalo, para cada una de las tres especies

```
lirios %>%
  group_by(Species) %>%
  summarise(across(contains('Petal'),
    list(Media = mean, DT = sd)))
```

## # A tibble: 3 × 5	## Species	Petal.Length_Media	Petal.Length_DT	Petal.Width_Media
	<fct>	<dbl>	<dbl>	<dbl>
## 1	setosa	1.4	0.0707	0.2
## 2	versicolor	4.54	0.336	1.44
## 3	virginica	5.68	0.356	2.1

Más información sobre el uso de `across()`

Cómo visualizar los datos

Esquema general

Vamos a usar `ggplot2`, otro paquete del [tidyverse](#)

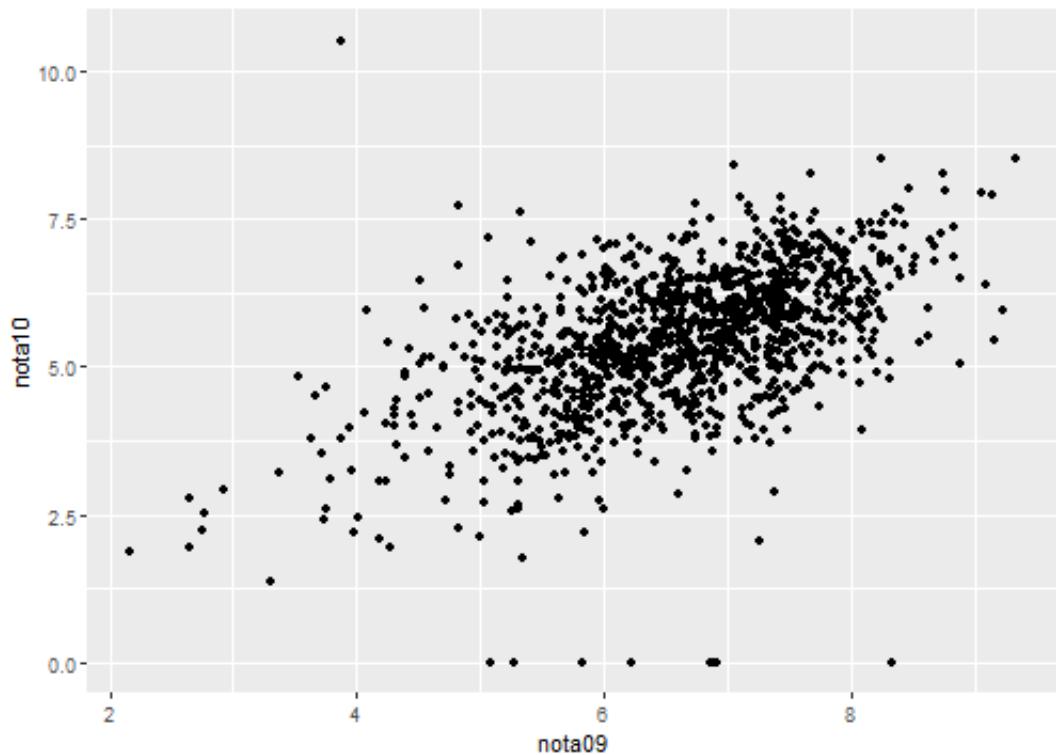
Este es el esquema general a seguir para crear un gráfico con `ggplot2`:

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

- `ggplot` crea un sistema de coordenadas vacío. El argumento `data` fija el `data frame` donde están los datos.
- `<GEOM_FUNCTION>` añade una capa al gráfico con el tipo de elementos que se van a representar.
- `mapping = aes(<MAPPINGS>)` asigna las variables del fichero a las propiedades visuales del gráfico.

Diagramas de dispersión

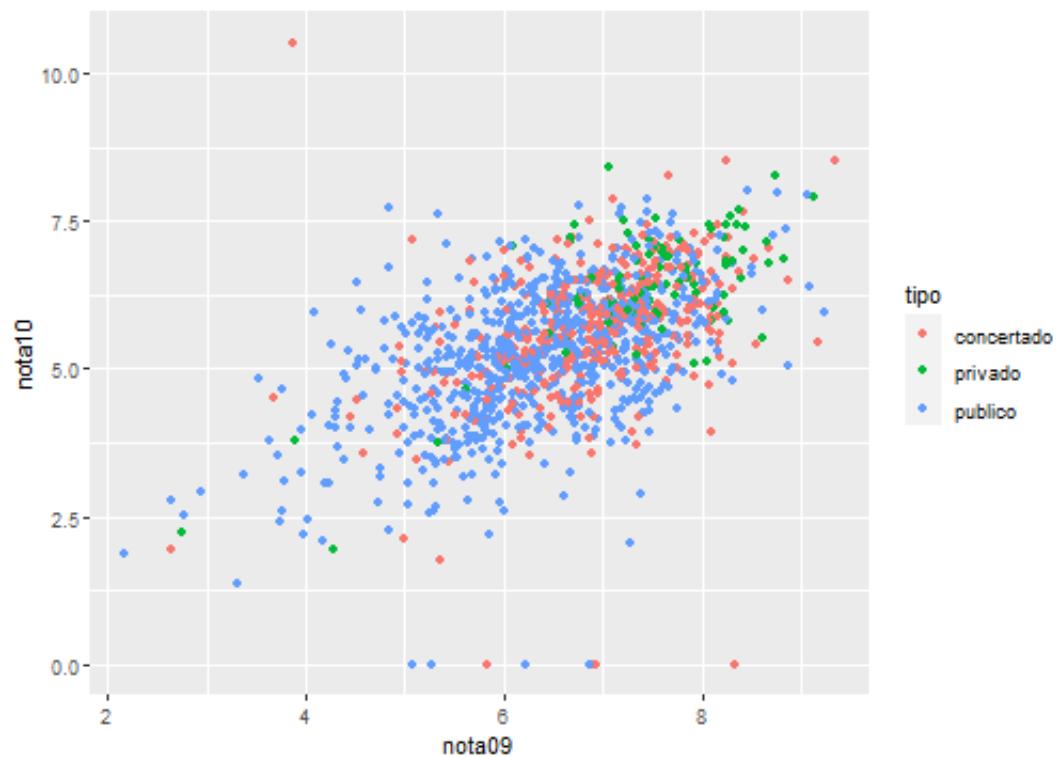
```
ggplot(data = notas) +  
  geom_point(mapping = aes(x = nota09, y = nota10))
```



Diagramas de dispersión

Asigna color según una variable cualitativa

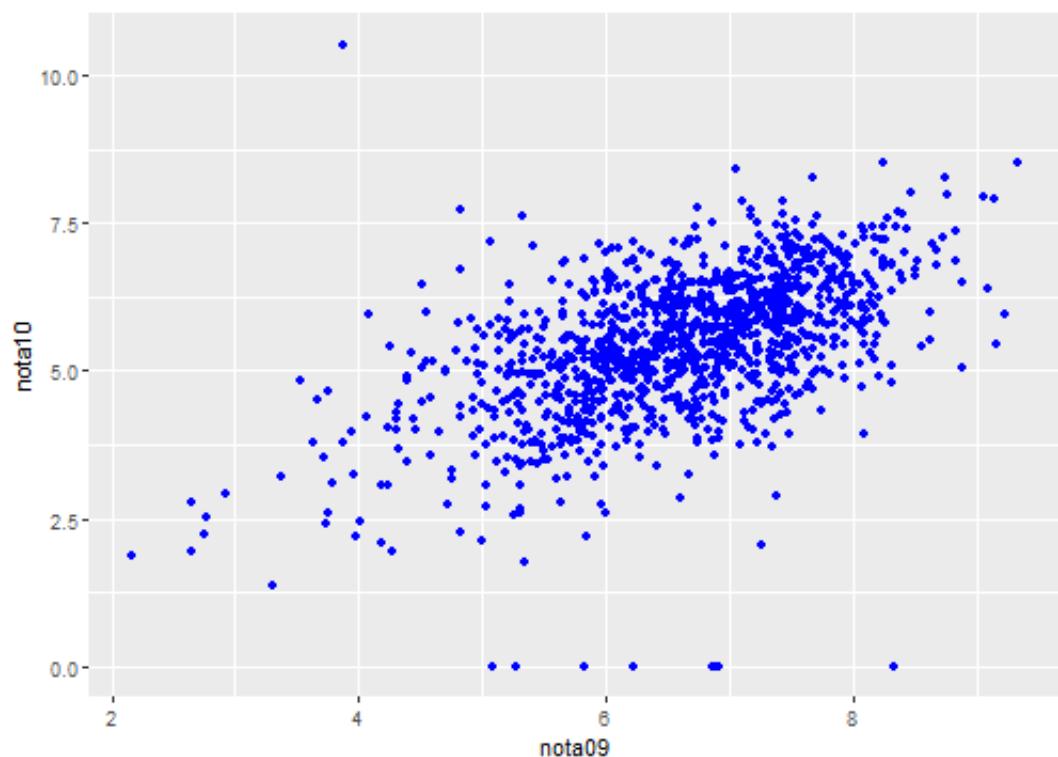
```
ggplot(notas) +  
  geom_point(aes(x = nota09, y = nota10, col = tipo))
```



Diagramas de dispersión

Compara con el ejemplo anterior

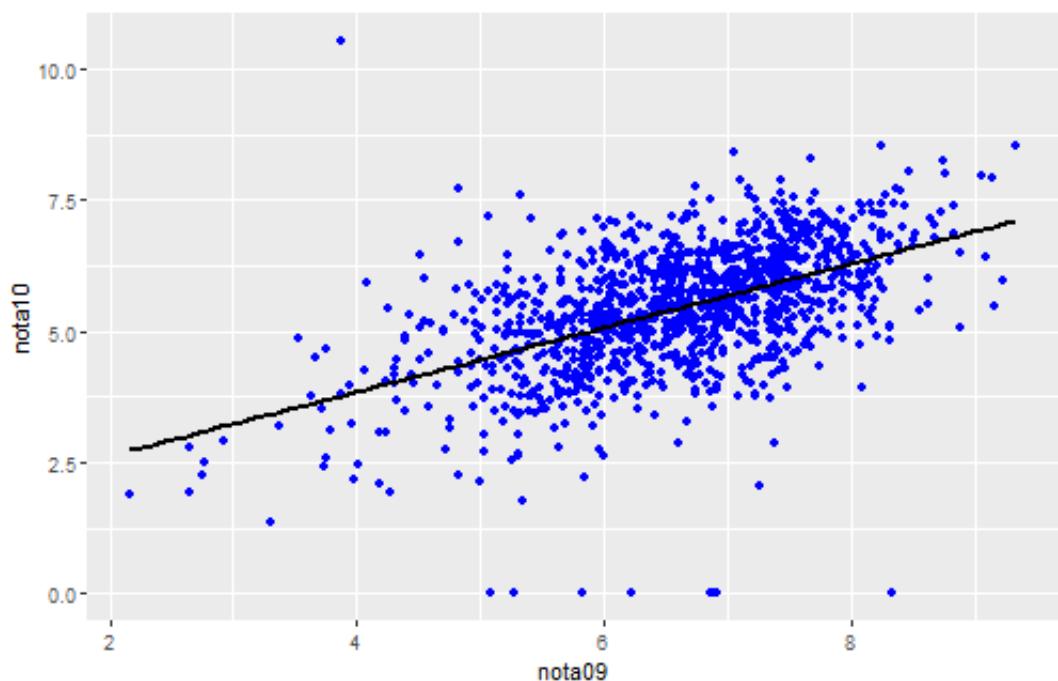
```
ggplot(notas) +  
  geom_point(aes(x = nota09, y = nota10), col = 'blue')
```



Diagramas de dispersión

Añade recta de mínimos cuadrados

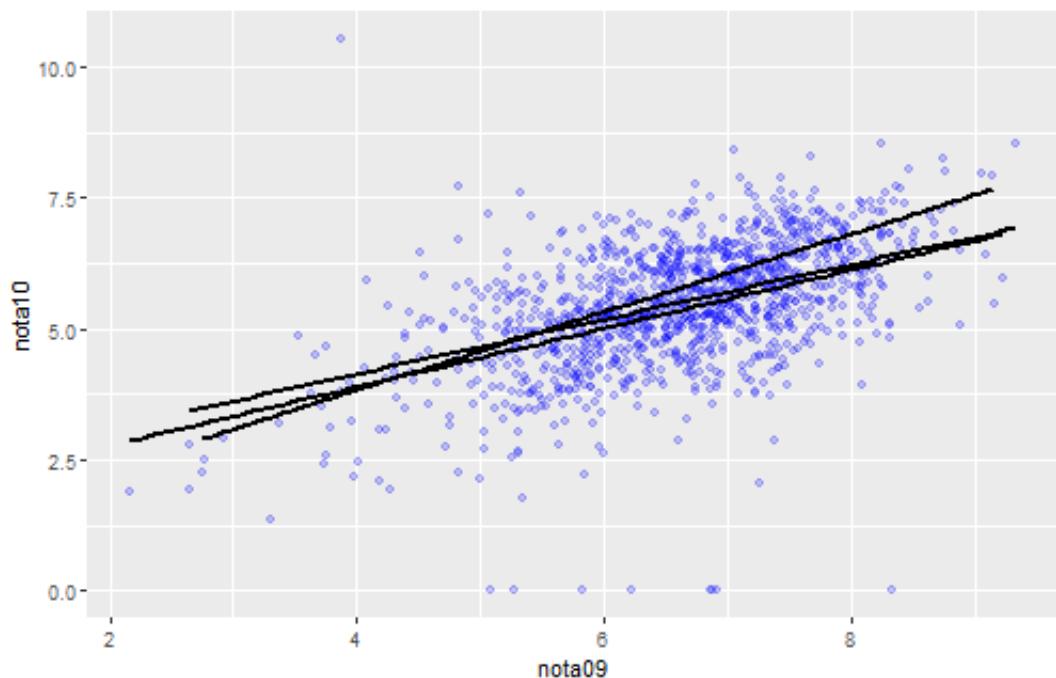
```
ggplot(notas, aes(x = nota09, y = nota10)) +  
  geom_point(col = 'blue') +  
  geom_smooth(method = 'lm', col = 'black', se = FALSE)
```



Diagramas de dispersión

Una recta para cada tipo de colegio

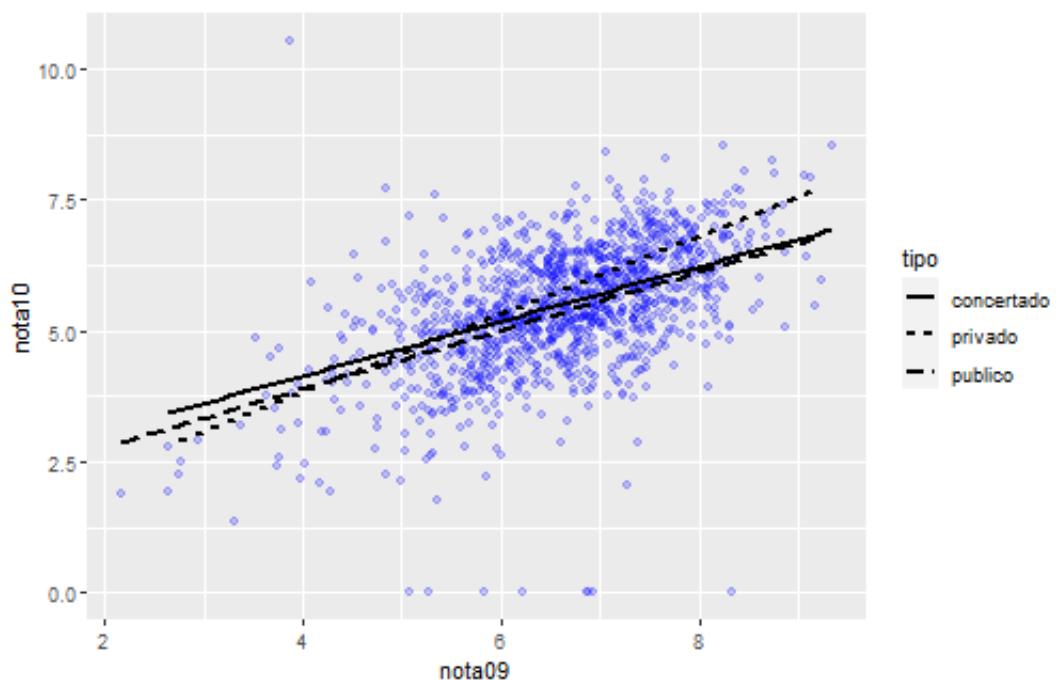
```
ggplot(notas, aes(x = nota09, y = nota10)) +  
  geom_point(col = 'blue', alpha = 0.2) +  
  geom_smooth(aes(group = tipo), method = 'lm',  
              col = 'black', se = FALSE)
```



Diagramas de dispersión

Compara con el ejemplo anterior

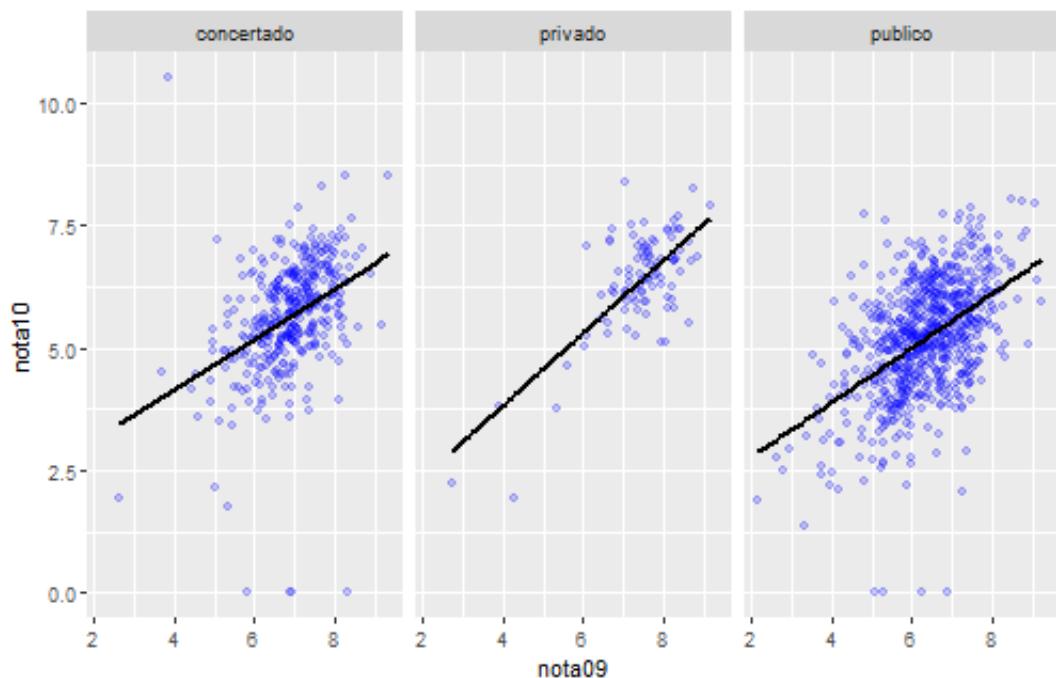
```
ggplot(notas, aes(x = nota09, y = nota10)) +  
  geom_point(col = 'blue', alpha = 0.2) +  
  geom_smooth(aes(linetype = tipo), method = 'lm',  
              col = 'black', se = FALSE)
```



Facets

Un gráfico para cada tipo de colegio

```
ggplot(notas, aes(x = nota09, y = nota10)) +  
  geom_point(col = 'blue', alpha = 0.2) +  
  geom_smooth(method = 'lm', col = 'black', se = FALSE) +  
  facet_wrap(~ tipo)
```

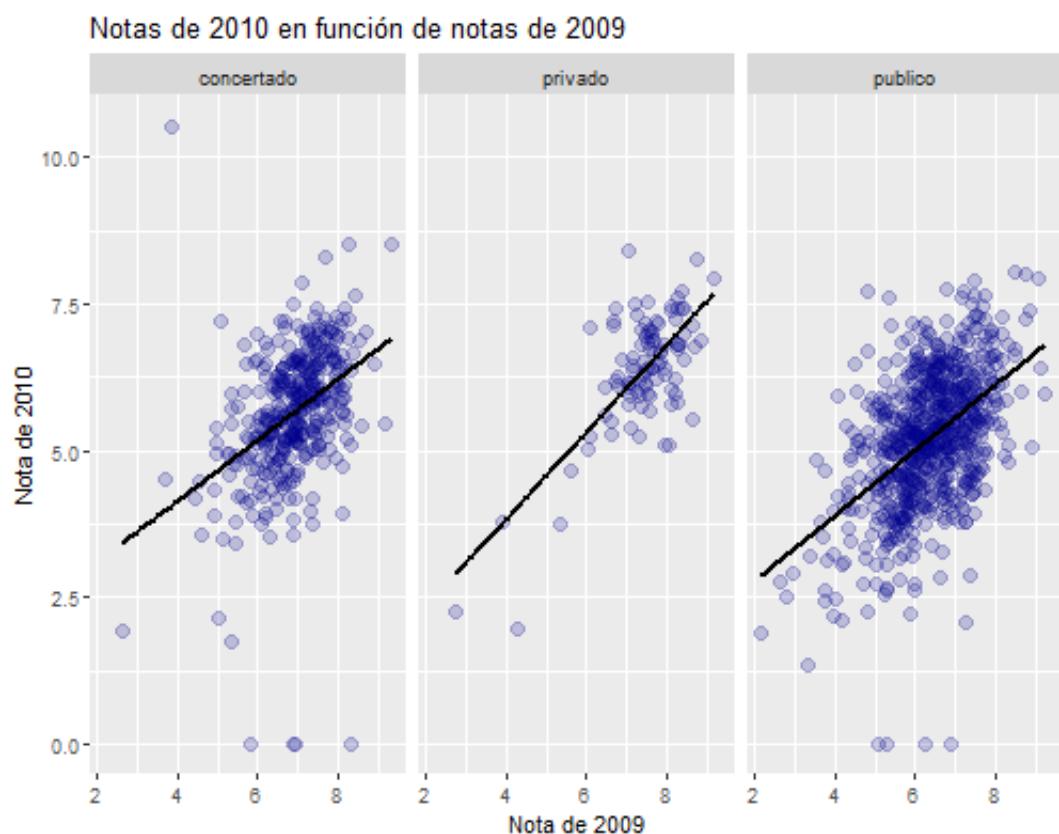


Títulos y etiquetas en los ejes

Podemos añadir un título o cambiar las etiquetas de los ejes de la siguiente forma:

```
ggplot(data = notas) +  
  geom_point(aes(x = nota09, y = nota10),  
             col = 'darkblue',  
             size = 3,  
             alpha = 1/5) +  
  geom_smooth(aes(x = nota09, y = nota10),  
              method = 'lm',  
              col = 'black',  
              se=FALSE) +  
  facet_wrap(~tipo) +  
  labs(title='Notas de 2010 en función de notas de 2009',  
       x='Nota de 2009',  
       y='Nota de 2010')
```

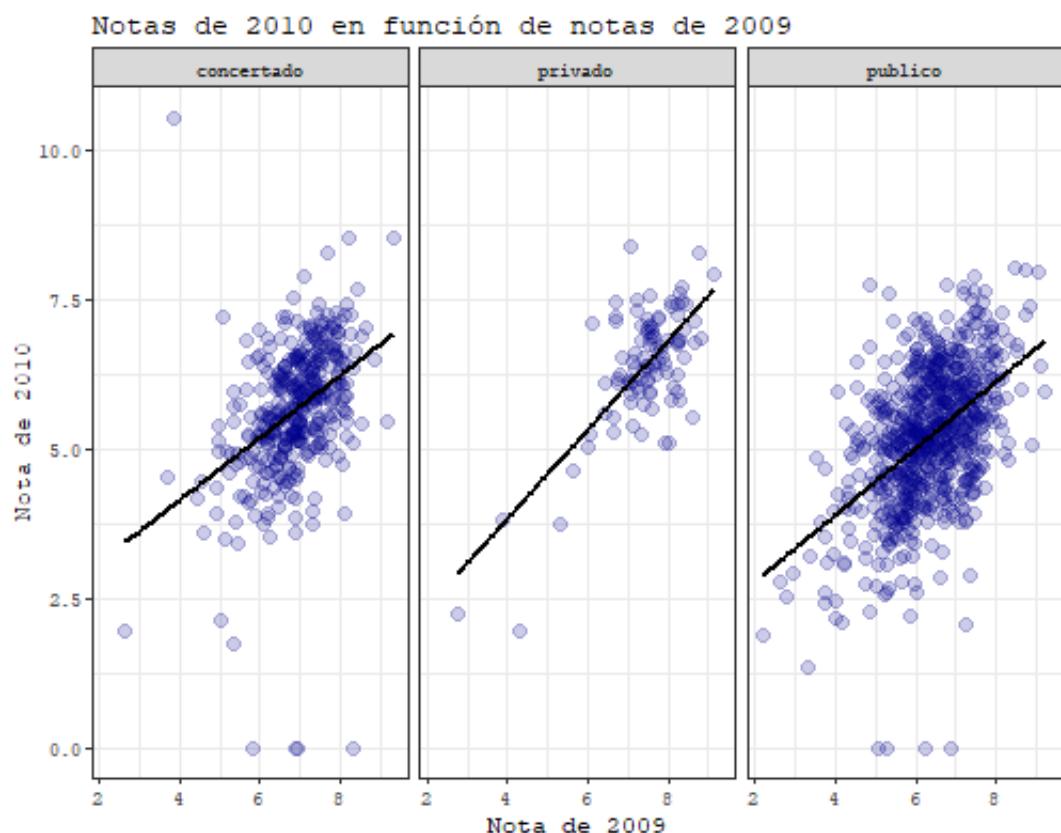
Títulos y etiquetas en los ejes



Aspecto general del gráfico (themes)

```
ggplot(data = notas) +  
  geom_point(aes(x = nota09, y = nota10),  
              col = 'darkblue',  
              size = 3,  
              alpha = 1/5) +  
  geom_smooth(aes(x = nota09, y = nota10),  
              method = 'lm',  
              col = 'black',  
              se=FALSE) +  
  facet_wrap(~tipo) +  
  labs(title='Notas de 2010 en función de notas de 2009',  
       x='Nota de 2009',  
       y='Nota de 2010') +  
  theme_minimal(base_family="mono")
```

Aspecto general del gráfico (themes)

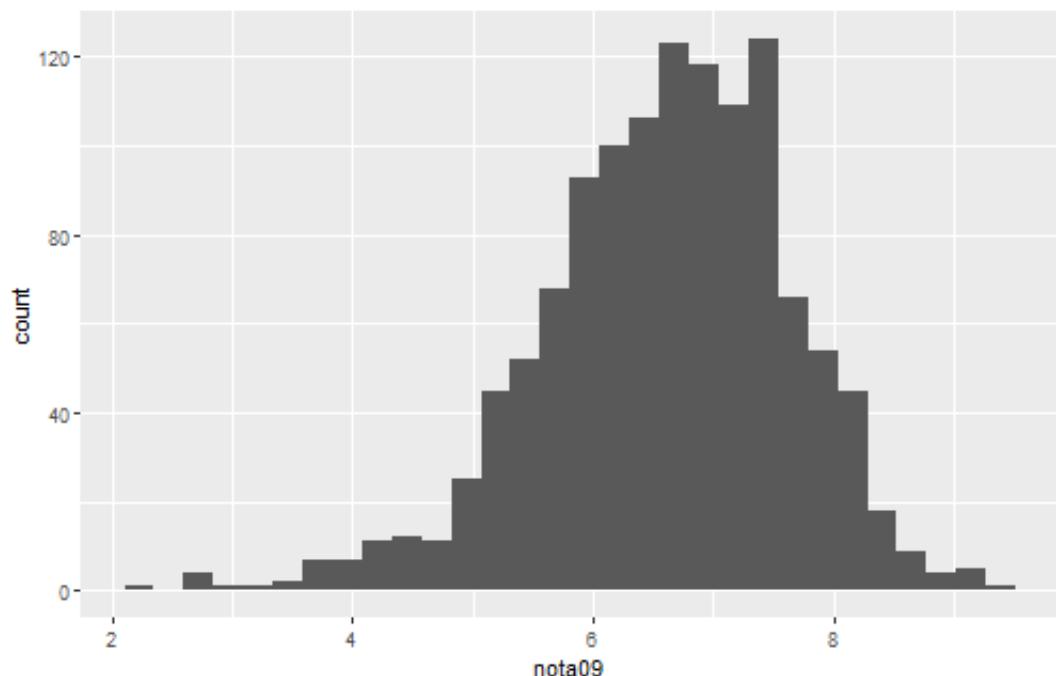


Más opciones para theme

Histogramas

Se usa `geom_histogram`

```
ggplot(data = notas) +  
  geom_histogram(aes(x = nota09))
```

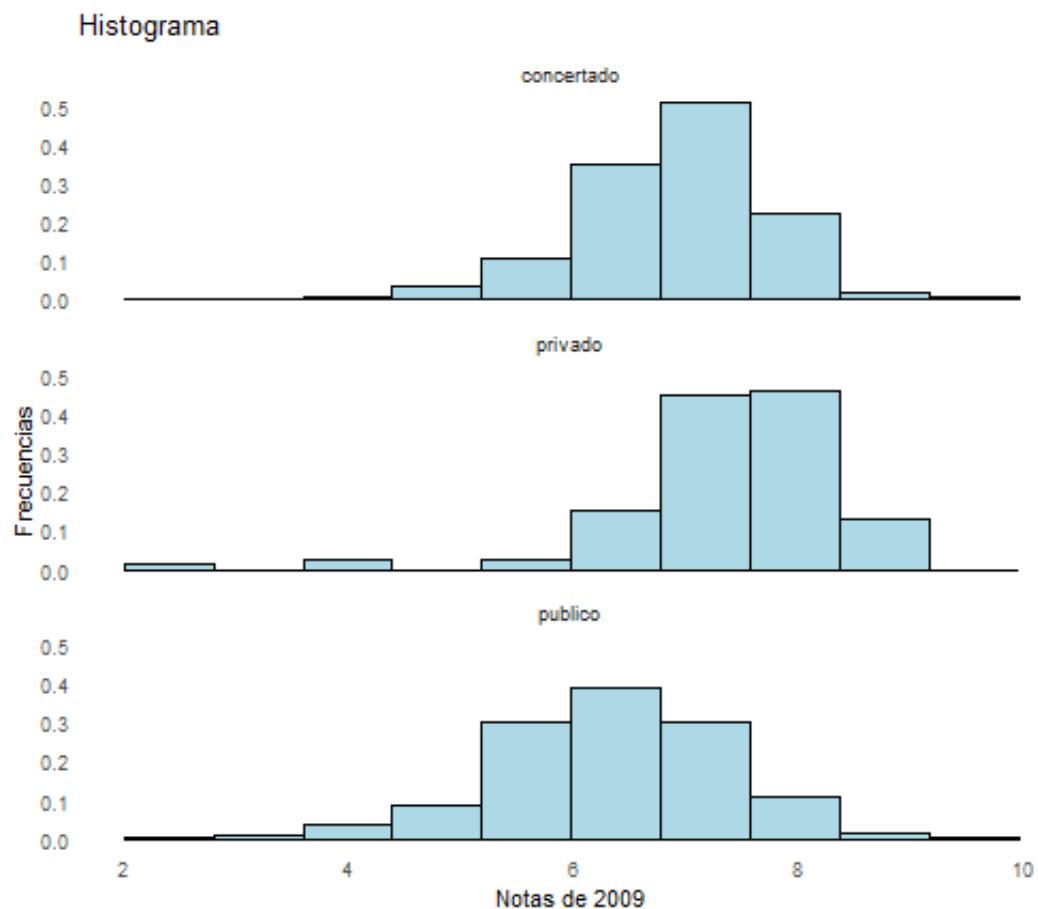


Histogramas

Cambia número de clases y otros aspectos del gráfico

```
ggplot(notas) +  
  geom_histogram(aes(x = nota09, y=..density..), # para qu  
                 bins=10, fill='lightblue', col='black') +  
  labs(x = 'Notas de 2009', y = 'Frecuencias', title = 'His  
facet_wrap(~tipo, nrow=3) +  
  theme_minimal() +  
  theme(panel.grid = element_blank()) # elimina grid de fo
```

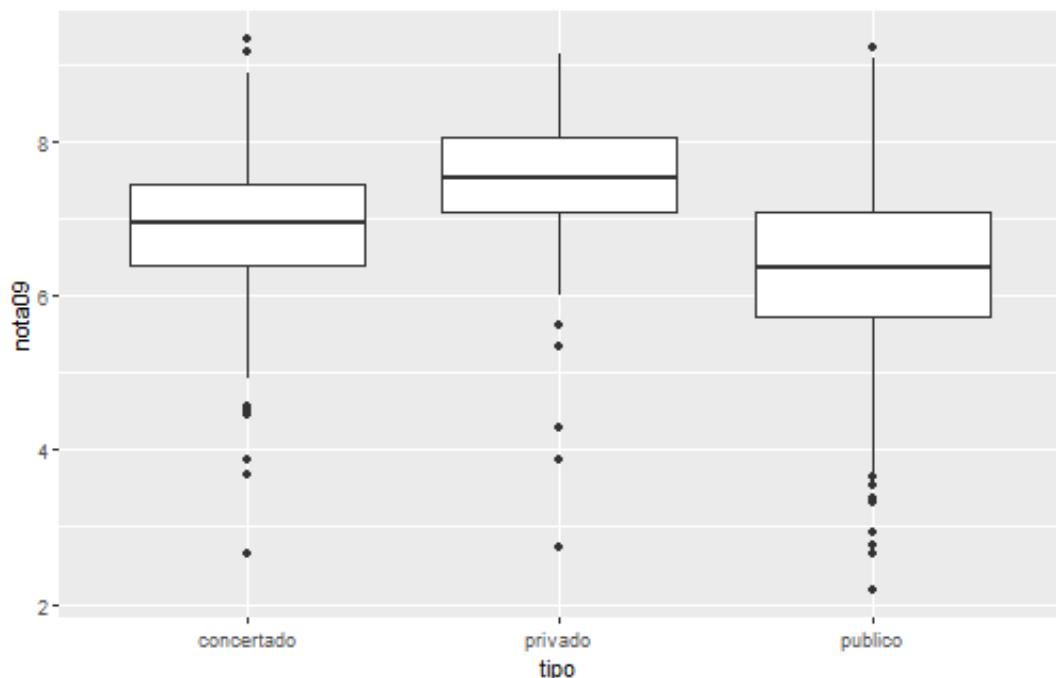
Histogramas



Diagramas de cajas

Se usa `geom_boxplots`

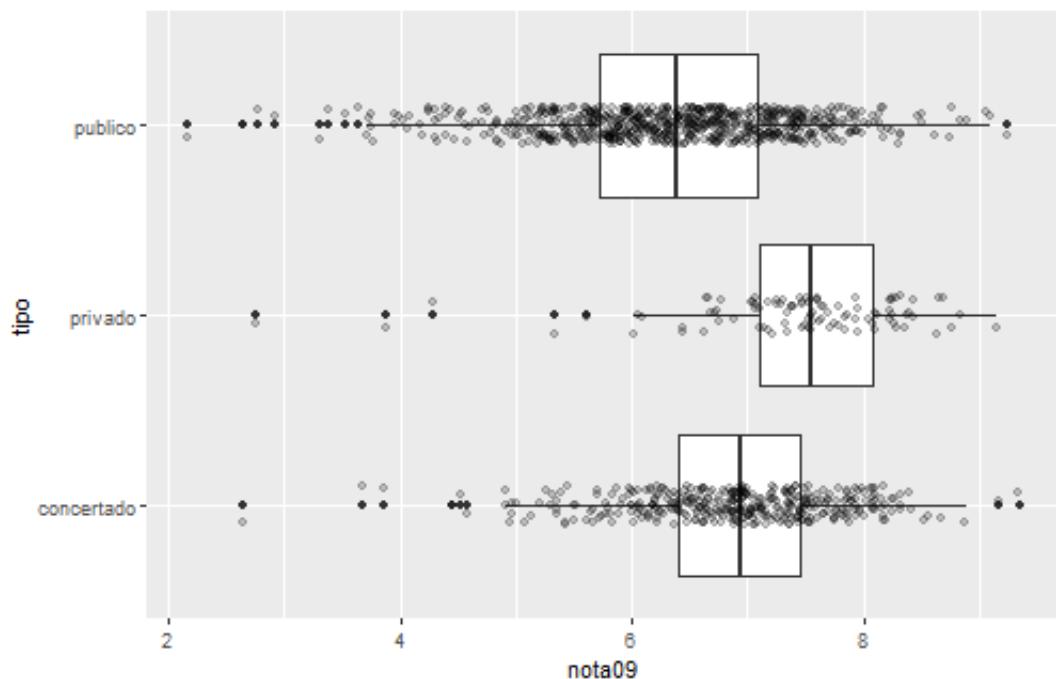
```
ggplot(notas, aes(x = tipo, y = nota09)) +  
  geom_boxplot()
```



Diagramas de cajas

Añade los puntos, intercambia los ejes

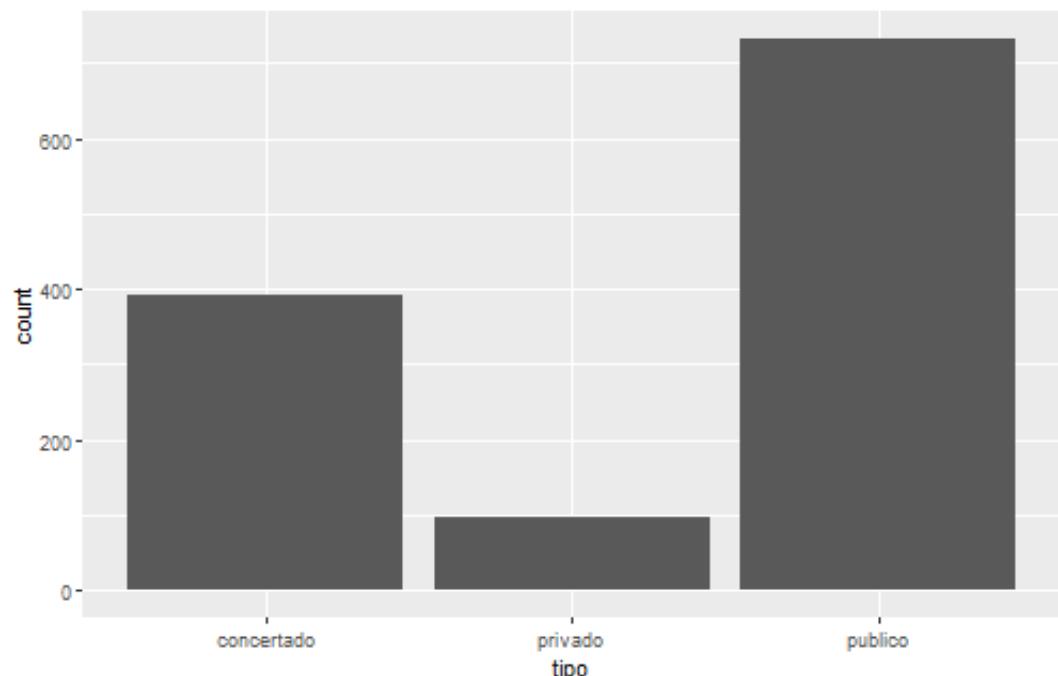
```
ggplot(notas, aes(x = tipo, y = nota09)) +  
  geom_boxplot() +  
  coord_flip() + geom_jitter(width = 0.1, alpha=0.2)
```



Gráficos de barras

Se usa `geom_bar`

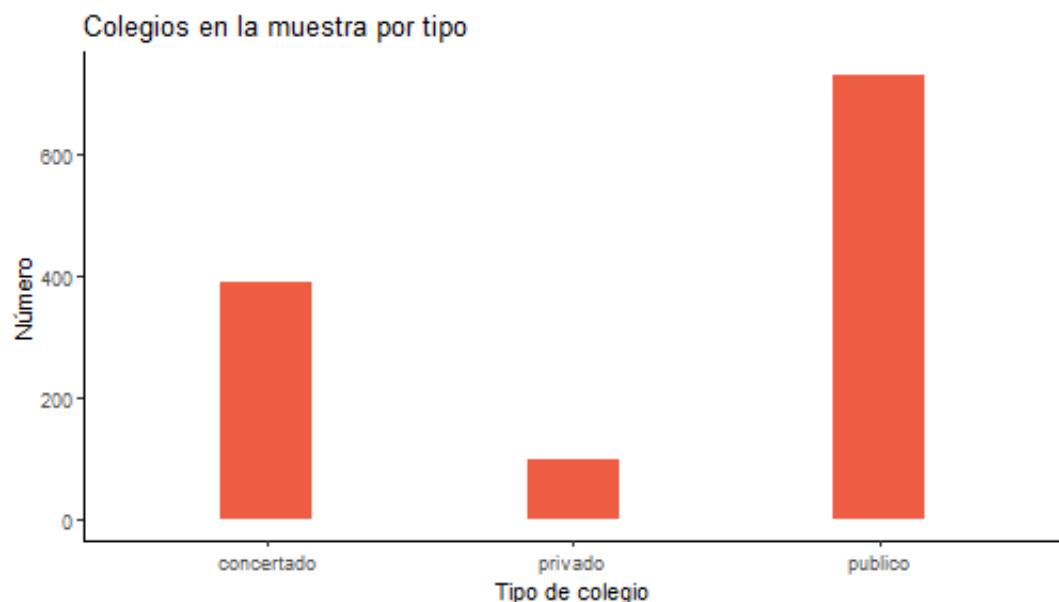
```
ggplot(data = notas, aes(x = tipo)) +  
  geom_bar()
```



Gráficos de barras

Otras opciones

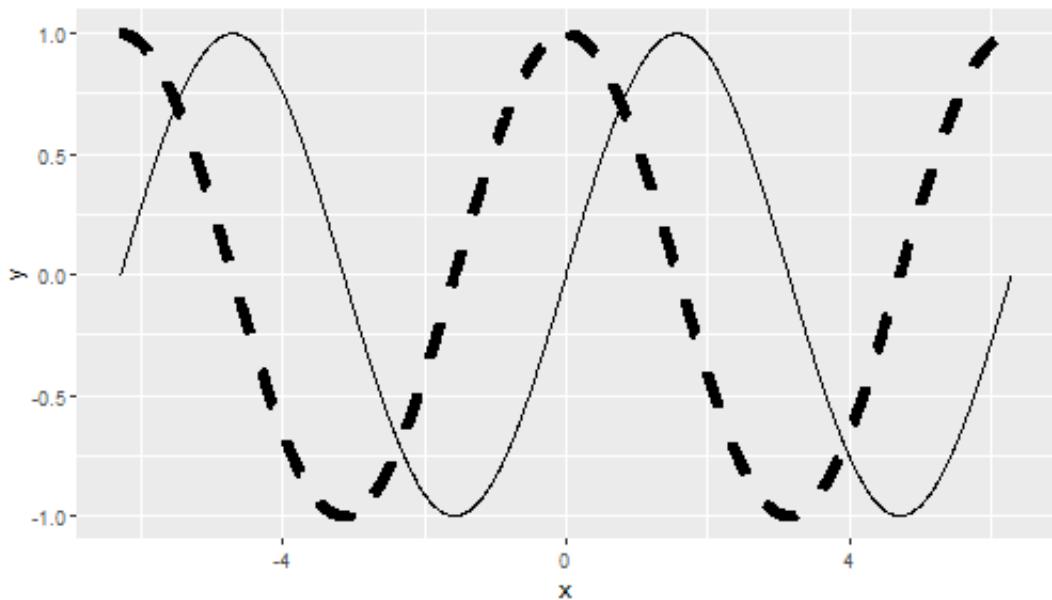
```
ggplot(data = notas, aes(x = tipo)) +  
  geom_bar(width = 0.3, fill='tomato2') +  
  labs(x='Tipo de colegio', y='Número', title='Colegios en  
  theme_classic()
```



Líneas

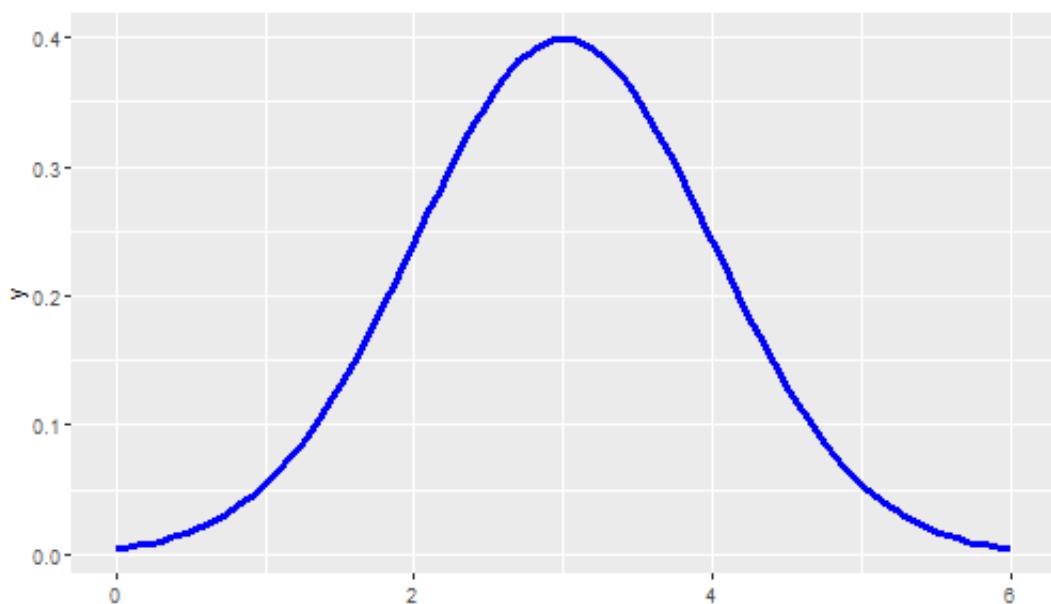
Se usa `geom_line`

```
x <- seq(-2*pi, 2*pi, 0.01)
datos <- data.frame(x = x, y = sin(x), z = cos(x))
ggplot(datos) +
  geom_line(aes(x=x, y=y)) + geom_line(aes(x=x, y=z), linet
```



Una función sin datos

```
ggplot() +  
  geom_function(fun = dnorm,  
                 args = list(mean = 3, sd = 1),  
                 size = 1.1, col = 'blue') +  
  xlim(0, 6)
```



Herramientas de comunicación

R Markdown

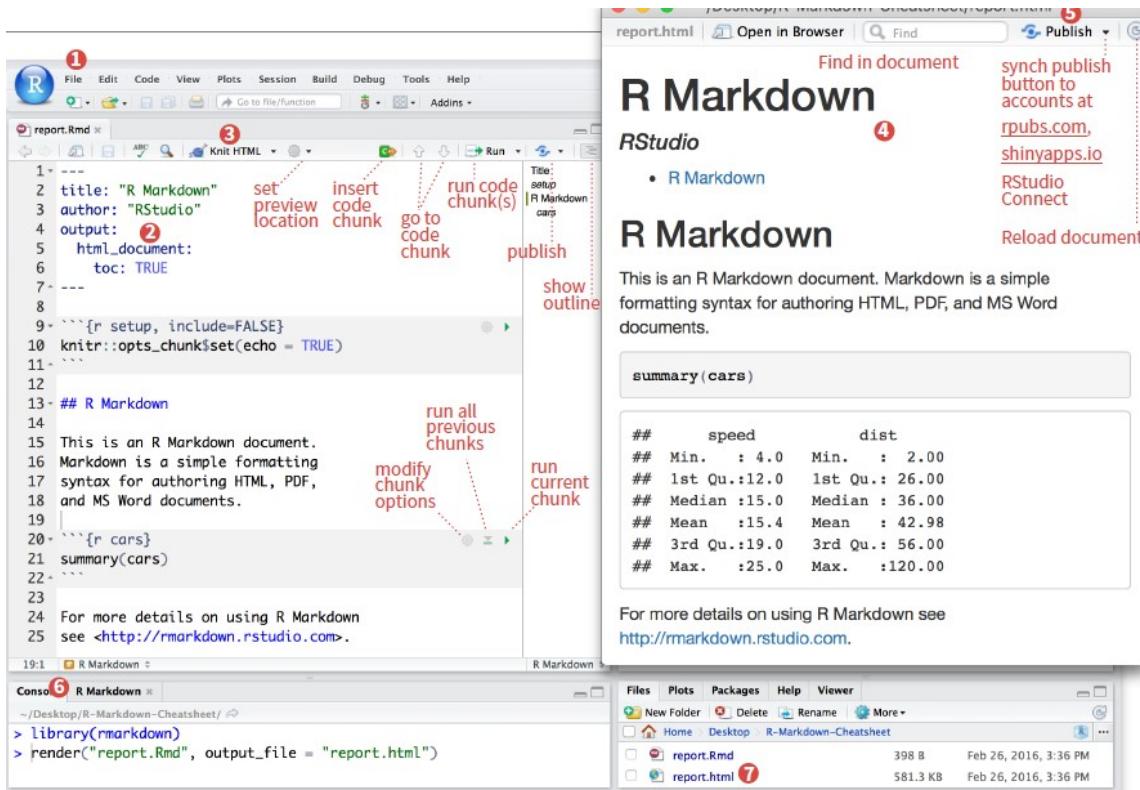
¿Qué es?

- Una colección de herramientas de comunicación en ciencia de datos.
- Un tipo de ficheros (.Rmd)
- Un paquete de R (rmarkdown)
- Un tipo de documento que integra texto, código y resultados

¿Para qué sirve?

- Comunicar las conclusiones de nuestro análisis
- Elaborar un proyecto en colaboración con otras personas
- Registrar no solo lo que hacemos en R, sino también lo que pensamos

Cómo trabajar con R Markdown



Cómo trabajar con R Markdown

1. Crear un nuevo fichero .Rmd
2. Escribir el documento editando el ejemplo
3. Usar el botón `Knit` para producir el informe
4. Visualizar el resultado
5. Publicar en la web
6. Notas técnicas en la consola
7. Documento final

Estructura de un fichero Rmarkdown (.rmd)

```
---
```

```
title: "Ejemplo"
author: "JRB"
date: "13/9/2019"
output: html_document
--
```

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
## R Markdown
```

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <<http://rmarkdown.rstudio.com>>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
```{r cars}
summary(cars)
```
```

- Encabezado optativo (*YAML header*) que incluye los *metadatos*
- Lineas de código de R (*chunks*)
- Texto (con formato *markdown*)

Metadatos

```
---
```

```
title: "Introducción a R"
subtitle: "11 y 12 de marzo de 2020, Universidad Autónoma de Madrid"
output:
  html_document:
    theme: yeti
    toc: true
    toc_float: true
---
```

Más opciones tecleando `?html_document` en la consola

Markdown

| | | |
|----------------------------|---|---|
| *Italic* | <u>Italic_</u> | <i>Italic</i> |
| **Bold** | <u><u>Bold</u></u> | Bold |
| # Heading 1 | Heading 1
===== | Heading 1 |
| ## Heading 2 | Heading 2
----- | Heading 2 |
| [Link](http://a.com) | [Link][1]
:
[1]: http://b.org | Link |
| ![Image](http://url/a.png) | ![Image][1]
:
[1]: http://url/b.jpg |  |
| > Blockquote | | Blockquote |

Markdown

- * List
- * List
- * List

- List
- List
- List

- List
- List
- List

1. One
2. Two
3. Three

- 1) One
- 2) Two
- 3) Three

1. One
2. Two
3. Three

'Inline code' with backticks

Inline code with backticks

...

```
# code block
print '3 backticks or'
print 'indent 4 spaces'
``
```

```
....# code block
....print '3 backticks or'
....print 'indent 4 spaces'
```

```
# code block
print '3 backticks or'
print 'indent 4 spaces'
```

Un curso interactivo de 10 minutos

Código

Opciones más usadas:

| Argumento | Efecto | Valor por defecto |
|-----------------|---|---------------------------------------|
| eval=FALSE | No ejecuta el código | TRUE |
| echo=FALSE | No incluye el código | TRUE |
| include=FALSE | Ejecuta el código pero no incluye ni código ni resultados | TRUE |
| warning=FALSE | No incluye los avisos | TRUE |
| message = FALSE | No incluye los mensajes | TRUE |
| fig.width | 7 | Anchura (en pulgadas) de los gráficos |
| fig.height | 7 | Altura (en pulgadas) de los gráficos |

Lista completa de opciones

Se puede insertar código dentro de una línea de texto

Referencias

Referencias

Bibliografía

- [Chang, W., R Graphics Cookbook](#)
- [Grolemund, G y Wickham, H, R for Data Science](#)

Otras páginas

- [La página sobre R Markdown en R Studio](#)
- [Un buscador de páginas relacionadas con R](#)
- [La página sobre R en mi blog | Mi blog](#)